Workshop

Nonlinear and Adaptive Approximation in High Dimensions

Bad Honnef, December 10–15, 2007

Tackling Higher Dimensions in Data Mining Using Adaptive Sparse Grids

Dirk Pflüger

Technische Universität München

Classification

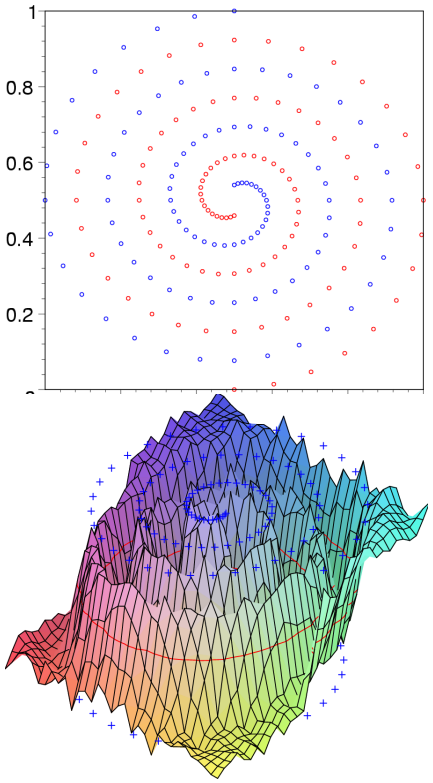
- Machine learning of a two-class problem
- Given:
  - Preclassified data set

$$S = \{(\mathbf{x}_i, y_i) \in [0,1]^d \times \{-1,1\}\}_{i=1}^{M}$$

  - Normalized data points $\mathbf{x}_i \in [0,1]^d$ with
  - Class labels $\mathbf{y}_i \in \{-1,1\}$
- Compute:
  - Classificator/Machine Learner (ML)

$$f : [0,1]^d \rightarrow \{-1,1\}$$

  - Provides class predictions applied on new data points



Regularization Network Approach

- classification as scattered data approximation problem
  + additional regularization terms

$$\text{minimize } H[f] = \frac{1}{M} \sum_{i=1}^{M} \mathcal{V}(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_K^2$$

- $\mathcal{V}$: cost/error function, e.g. $(y_i - f(\mathbf{x}_i))^2$
- $\|f\|_K^2$: regularization operator/stabilizer, e.g. $\|\nabla f\|_{L_2}^2$
- λ: regularization parameter

- Minimize trade-off between costs and smoothness
- Different Neural Networks and Support-Vector-Machines can be formulated as Regularization Network Approach

Discretization – Why Sparse Grids?

- Discretization of feature space not feasible: Curse of dimensionality!
  - $\tilde{N}$ grid points in 1-d $\rightarrow \tilde{N}^d$ grid points in d-d
- Common classification algorithms:
  mostly global ansatz functions, associated to data points
  - Good to reduce number of basis functions
  - But problem for massive sets of training data:
    $\mathcal{O}(M^2)$ or worse!
- Aim: classification algorithm with $\mathcal{O}(M)$
  - Useful e.g. for automatically collected training data
- Can be achieved with sparse grids
  - Similar accuracy
  - Only $\mathcal{O}(\tilde{N} \cdot \log(\tilde{N})^{d-1})$ basis functions!

Spatial Discretization

- Idea: introduce some degree of data-independency
- Discretize space, apply ansatz functions associated to grid points
- Introduce basis $\{\phi_i\}_{i=1}^{N}$
- Restrict problem to finite dimensional space V_N spanned by basis functions,
  e.g. space of piecewise d-linear functions:

$$f_N(\mathbf{x}) = \sum_{j=1}^{N} \alpha_j \phi_j(\mathbf{x})$$

- Minimization leads to linear system with N unknowns:

$$\left(\lambda MC + B \cdot B^T\right) \boldsymbol{\alpha} = B\mathbf{y}$$

with

$$C_{ij} = \left(\nabla\phi_i(\mathbf{x}), \nabla\phi_j(\mathbf{x})\right)_{L_2}$$

$$B_{ij} = \phi_i(\mathbf{x}_j)$$

- Solve, for example, with CG

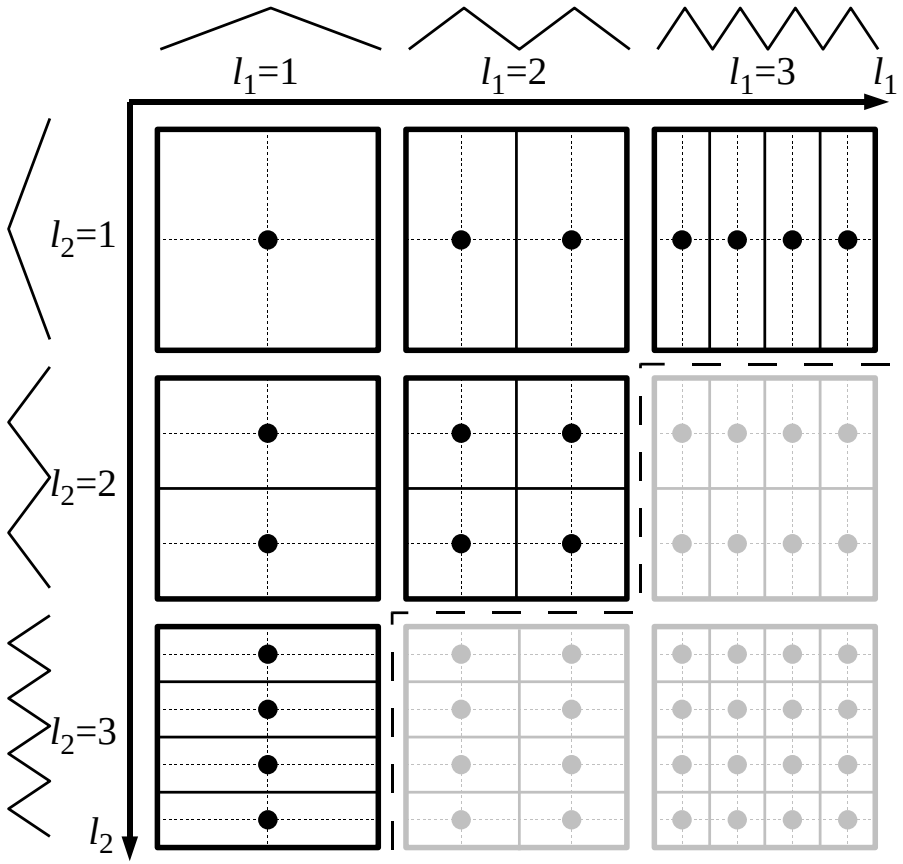
Sparse Grids

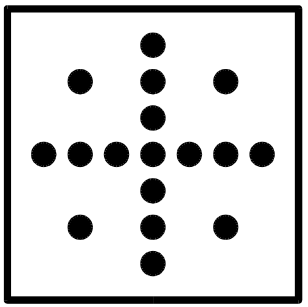
- Developed for solution of PDEs
- Applied successfully to
  - Numerical quadrature
  - Interpolation
  - Approximation
  - Data storage
  - Optimization
  - ...

Sparse Grids

- Sparse grid space $V_n^{(1)}$:

$$V_n^{(1)} := \bigoplus_{|\mathbf{l}|_1 \leq n+d-1} W_{\mathbf{l}}$$

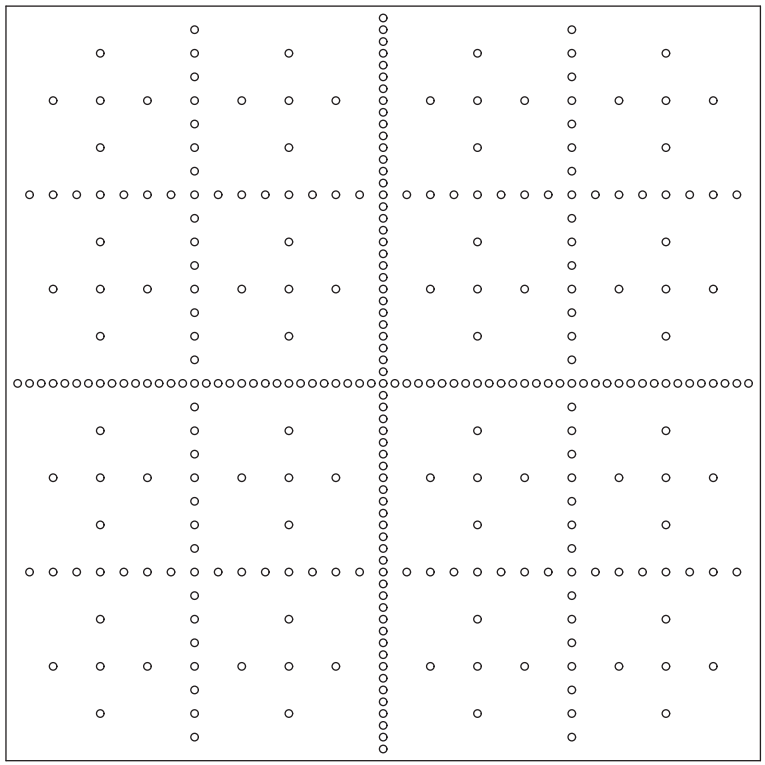


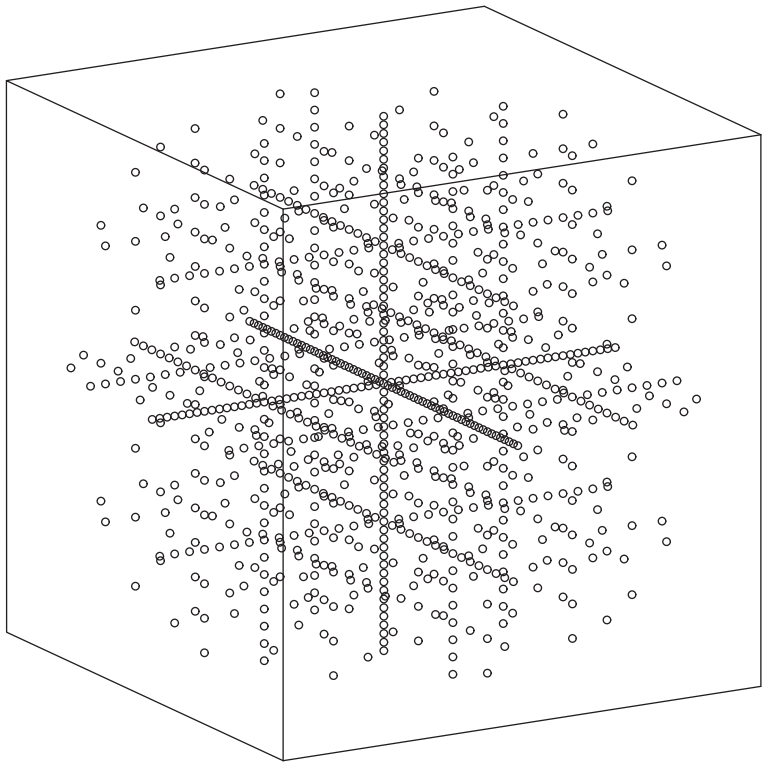


$V_3^{(1)}$

Sparse Grids, Examples

- Optimal selection of subspaces leads to sparse grid space $V_n^{(1)}$
- Examples for underlying sparse grids for level 6 in 2 and 3 dimensions



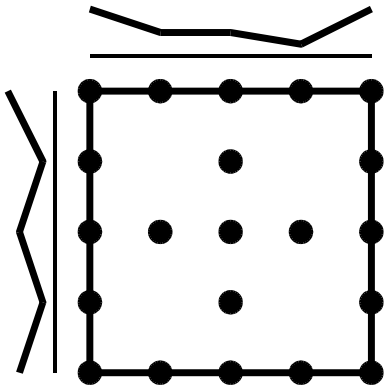


Sparse Grids – Two Approaches

- Combination technique:
  Multivariate extrapolation scheme, superposition of coarser reg. grids
  - Multiple, but smaller and regular grids
    ($\mathcal{O}(d \cdot l^{d-1})$ problems of size $\mathcal{O}(2^l)$)
  - Efficient solvers can be used
  - Parallelization straightforward

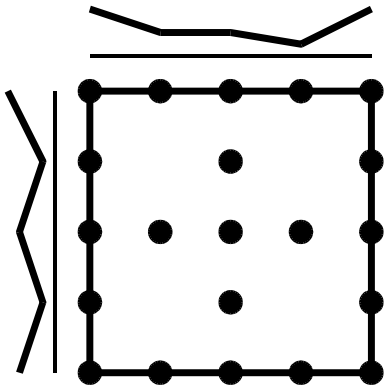
- Direct adaptive technique:
  Direct solution in sparse grid space
  - One, but larger problem
    (one of size $\mathcal{O}(2^l \cdot l^{d-1})$)
  - Allows for adaptivity!

Boundary considerations

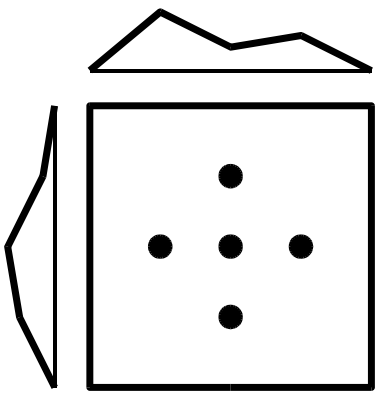


- Rather than $\mathcal{O}(\tilde{N}^d)$ only $\mathcal{O}(\tilde{N} \cdot log(\tilde{N})^{d-1})$ unknowns
- Still: many grid points on boundary
  - For just one inner point $\mathcal{O}(3^d)$ grid points
  - Limits dimensionality because of storage!
- Better: without
  - DM: often smooth near boundary
  - Adaptivity

Boundary considerations



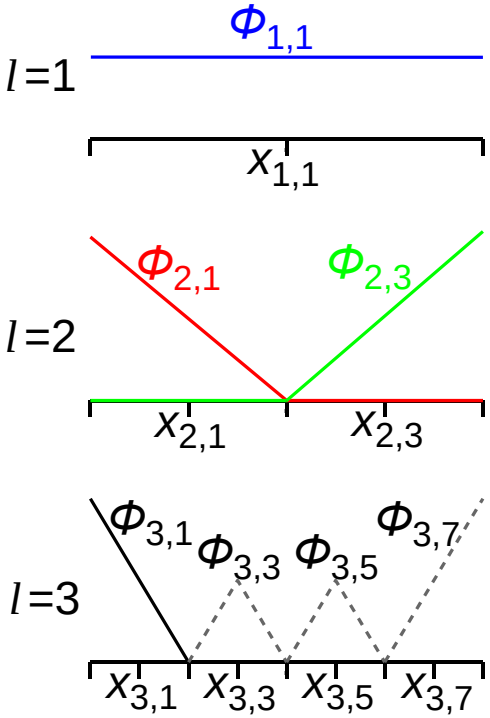
- Rather than $\mathcal{O}(\tilde{N}^d)$ only $\mathcal{O}(\tilde{N} \cdot log(\tilde{N})^{d-1})$ unknowns
- Still: many grid points on boundary
  - For just one inner point $\mathcal{O}(3^d)$ grid points
  - Limits dimensionality because of storage!
- Better: without
  - DM: often smooth near boundary
  - Adaptivity



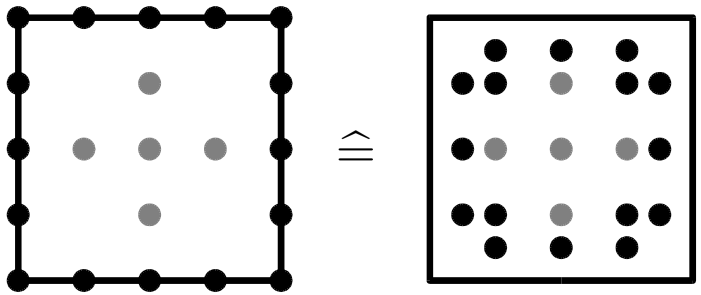
Therefore:

- Normalize data to slightly smaller region
- Use modified basis functions near boundary
- Start with only $3d + 1$ grid points

Modified Boundary Functions



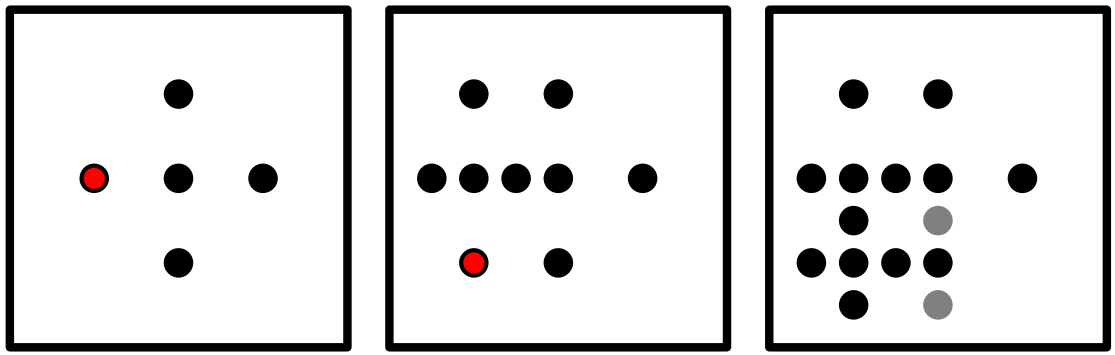
- Adaptivity + modified boundary functions makes basis functions on boundary obsolete



- Main result: not necessary for moderate dimensionalities

Adaptive Sparse Grids

- Sparse grids are intrinsically adaptive
- Start with $V_2^{(1)}$, refine grid point with highest surplus



- Refine only where necessary
- Reasonable:
  - Real data clustered together, steep and flat regions
  - Too many grid points at wrong place: overfitting

Solving the System of Linear Equations

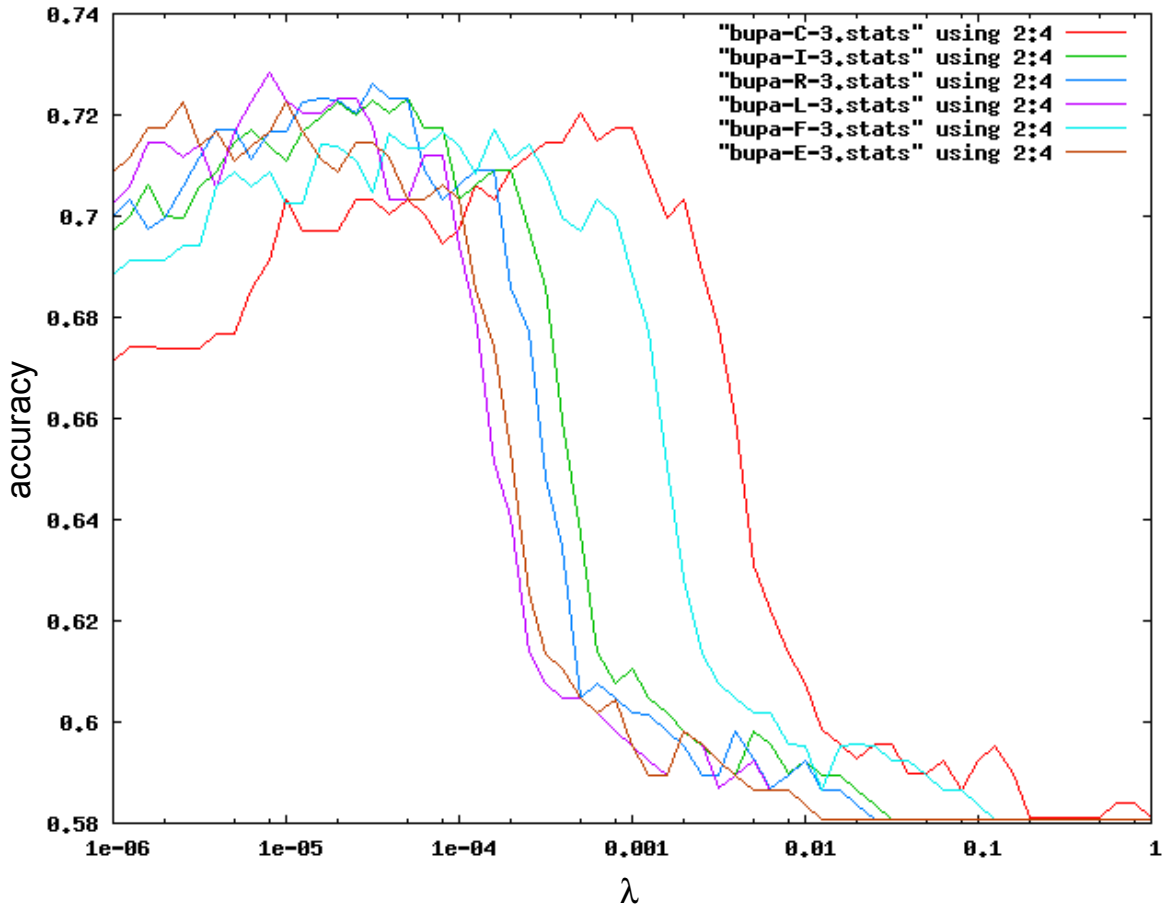
- $A := \left(\lambda MC + BB^T\right)$ not sparse
- Solve iteratively
- $B_{ij} = \phi_i(\mathbf{x}_j)$ simple
- "Classical" $C_{ij} = \left(\nabla\phi_i(\mathbf{x}), \nabla\phi_j(\mathbf{x})\right)_{L_2}$ algorithmically challenging
  - *UpDown*-algorithm: traversal of "tree" of basis functions

$$C_{ij} = \sum_{s=1}^{d} \left(\nabla\phi_{i_s}(x_s), \nabla\phi_{j_s}(x_s)\right)_{L_2} \prod_{t\neq s} \left(\phi_{i_t}(x_t), \phi_{j_t}(x_t)\right)_{L_2}$$

  - Linear in N
  - But – $\mathcal{O}(d \cdot 2^d \cdot N)$ operations – depends exp. on d
- Regularization operator $\|f\|_K^2$ bottleneck!

Regularization Operator

- Comparison of different matrices C
- $C = \Delta$, $C = I$, C depending on support, ...



Regularization Operator (2)

- Similar behaviour (apart from scaling λ), performance dependent on dataset
- Usage of I
  - $\mathcal{O}(d \cdot 2^d N) \Rightarrow \mathcal{O}(N)$
  - Corresponds to regularization operator

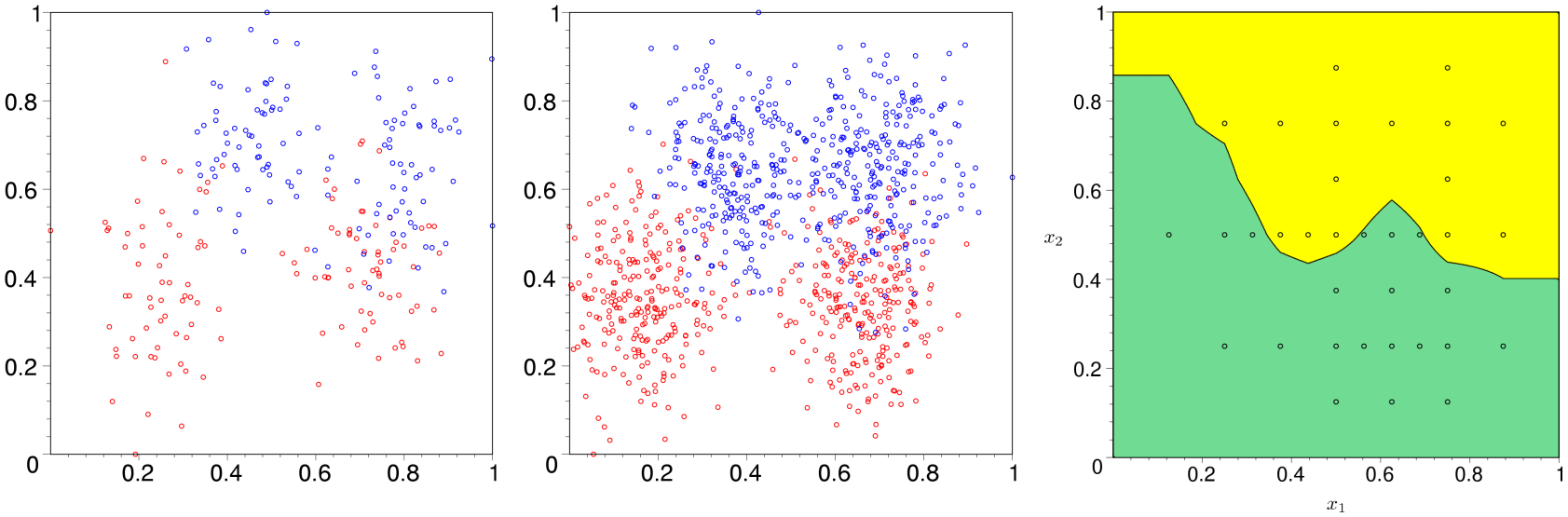
$$\|f\|_K^2 = \frac{1}{2} \sum_{j=1}^{N} \alpha_j^2 \quad \text{rather than} \quad \|\nabla f\|_{L_2}^2$$

  - Independent of support
  - But smaller support: less influence in $\mathcal{V}(y_i, f(\mathbf{x}_i)) = (y_i - f(\mathbf{x}_i))^2$

$$H[f] = \frac{1}{M} \sum_{i=1}^{M} \mathcal{V}(y_i, f(\mathbf{x}_i)) + \lambda \|f\|_K^2$$

Example: Ripley Dataset

- Training (250) and test data (1000), 8% noise



- Most refinement in critical region
- Accuracy on test data 91.5%

Example: Bupa Liver Dataset

- Bupa liver dataset (UCI repository)
- Real vital data of 345 patients for liver illness, 6D
- $\lambda = 0.01$, 10-fold testing results

adapt. sg $\lambda = 0.01$			comb. techn. lin. anisotrop.	SVM linear	SVM non-linear
# refinements	|grid|	acc. [%]	acc. [%]	acc. [%]	acc. [%]
7	403	72.22	73.9	70.0	73.7
13	1091	74.61			
15	1371	76.30			

- Sparse grid techniques: competitive performance
- Here: adaptivity additional benefit

Tackling Even Higher Dimensionalities

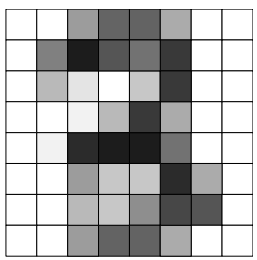
- Status:
  - Storage $\mathcal{O}(N)$, start with $N = 2d + 1$
  - Complexity $\mathcal{O}(dN)$ (one CG-step)
- Enables to tackle high dimensional problems
  - But: most datapoints close to the boundary, normal hat basis functions fail
  - Solution: use modified boundary functions again

High Dim Examples

Sonar, 60D

- Radar signals, mines vs. rocks
- 208 data points, 10-fold
- Best result: 92.78%, $\lambda = 0.0046$, 15 refinements, 3110 grid points
- Time on std. PC: 190min
- Best other: NN, 90.04%

Optical recognition of handwritten digits, 64D



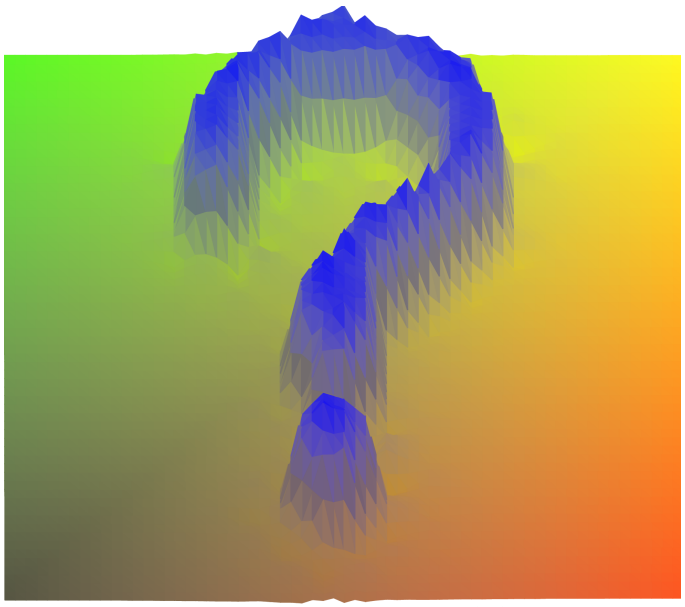
- 8x8 pattern
- 3823 training, 1797 testing
- One classifier for each class
- Results: 98.87–99.94%, dep. on class (1–8 refinements)
- Time on std. PC: 4min–120min
- Compared to: k-NN: 98.00%, Tree-SVM: 85.41%

Summary and Current Work

- Sparse Grids provide universal, flexible classification method
  - Linear in number of training data
- Adaptivity exploits structure of data very well
  - Provides trade-off between data dependency and data independency
  - Refinement steep in only few dimensions (dimension adaptivity)
  - Modifications allow tackling higher dimensions

- Examine dimension adaptive properties more thoroughly
- Further investigation of regularization operator

Thank you for your attention!



$$\lambda = 0.001, \quad V_8^{(1)}$$